Almost-Everywhere Superiority for Quantum Polynomial Time[1]

Edith Hemaspaandra

Department of Computer Science, Rochester Institute of Technology, Rochester, New York 14623
E-mail: eh@cs.rit.edu

Lane A. Hemaspaandra

Department of Computer Science, University of Rochester, Rochester, New York 14627
E-mail: lane@cs.rochester.edu

and

Marius Zimand

Department of Computer and Information Sciences, Towson University, Towson, Maryland 21252; and Department of Computer Science, University of Bucharest, Bucharest, Romania
E-mail: mzimand@saber.towson.edu

Simon as extended by Brassard and Høyer shows that there are tasks on which polynomial-time quantum machines are exponentially faster than each classical machine infinitel often. The present paper shows that there are tasks on which polynomial-time quantum machines are exponentially faster than each classical machine almost everywhere.

Key Words: quantum computing; almost-everywhere superiority; probabilistic computing; computational complexity.

1. INTRODUCTION

One issue of broad importance in the area of quantum computing is to gain an understanding of exactly what potential quantum computers hold, i.e., what superiority over classical computers they offer. Work of Simon [18], as extended by Brassard and Høyer [8],[2] is often cited as evidence for the potential superiority of quantum computation over classical computation. Their work shows that for computation relative to a black-box function (also sometimes referred to as a promise function) there are problems on which exact (i.e., worst-case polynomial time with zero error probability, see, e.g., [8]) polynomial-time quantum computation is infinitel often exponentially faster than each deterministic—or even bounded-error probabilistic—classical computer solving the problem.

Berthiaume and Brassard [6] raised and studied the issue of whether one can obtain far more decisive separations: separations where quantum computation is superior on all but a finit number of inputs. That is, they sought to bring to the quantum-versus-classical computation question the very strong type of separation known in complexity theory as almost-everywhere separations [1, 10, 11]. Berthiaume and Brassard [6] obtained the remarkable result that there are tasks that can be done in exact exponential time on quantum machines but on which each classical deterministic machine requires double exponential time almost everywhere.

However, neither the Berthiaume–Brassard result nor the technique of its proof works for quantum polynomial time. (Such results tend to be far harder to obtain for small time classes than for large ones.) Also, their result is for deterministic (not bounded-error probabilistic) classical computing. Additionally, the Simon result leaves open the possibility (which in fact is the case that holds, as firs discussed by Berthiaume and Brassard) that for some classical computers solving the problem there are infinitel many inputs on which quantum computing is not interestingly faster on these problems. In fact, quantum

[1] Supported in part by grants NSF-CCR-9322513 and NSF-INT-9815095/DAAD-315-PPP-gü-ab, and an RIT FEAD grant.
[2] An alternative algorithm to that of Brassard and Høyer was obtained by Mihara and Sung [15] and by Beals [14].

0890-5401/02 $35.00

computing in Simon's construction is superior of classical computing on only a logarithmically small portion of the inputs.

In contrast, the present paper shows that there are problems on which exact quantum polynomial-time computing is exponentially superior to classical computing almost everywhere. In particular, we show that for computation relative to a black-box function there are problems solved in exact polynomial-time by quantum computers but on which every deterministic—or even bounded-error probabilistic—classical computer solving the problem requires exponential time on all but a finite number of inputs.

2. HISTORY AND DISCUSSION

This section provides a more detailed history and discussion of the background and related results than does Section 1. Reading this section is not needed to understand the results of Section 3.

2.1. Simon and Infinitely-Often Superiority for Quantum Computing

Tremendously exciting new models of computation—quantum computing and DNA-computing—have become one strong focus of theoretical computer science research. Researchers dearly want to know whether these models, at least in certain settings, offer computational properties (most particularly, quick run-time) superior to what is offered by classical computational models.

Of course, even such an exciting model as quantum computing has limitations (see, e.g., the elegant lower-bound approach of Beals et al. [4]). However, let us here consider the highlights of what is known suggesting the superiority of quantum computing. The three most famous lines of work are those of Shor [17], Grover ([12], see also [7]), and Simon ([18], see also [8]).

Grover shows that quantum computing can solve certain search problems at a quadratically faster speed than one intuitively would expect in classical computing. Shor shows that factoring (and other interesting problems) can be performed in expected polynomial time in the quantum model; this is a superpolynomial speedup with respect to the best currently known classical solutions. These are both undeniably impressive results. However, note that it is at least plausible that classical, deterministic computing can seemingly have the effect of searching through huge numbers of possibilities very quickly (for example, testing satisfiability) and can factor very quickly. For example, if $P = NP$, NP-like search problems[3] are easily in P, though clearly not by going through all the possibilities in anywhere near a brute-force way; even if $P \neq NP$, it might still be the case that deterministic polynomial-time algorithms exist for factoring.

In contrast, Simon [18] shows that for computing with respect to a black-box function there are problems for which quantum polynomial-time bounded-error computing provably *is* exponentially faster than classical deterministic computing or even classical bounded-error computing. Brassard and Høyer [8] improved the upper bound to obtain that for computing with respect to a black-box function there are problems for which exact quantum polynomial-time computing is exponentially faster than classical deterministic computing or even classical bounded-error computing; in particular, there are problems in exact polynomial time (which we will refer to as QP, see [8], though it sometimes is denoted EQP) such that each bounded-error classical Turing machine solving them requires exponential time on infinitely many inputs.

[3] To be fair to Grover, his result can plausibly be viewed instead as a black-box result. The key issue is whether the predicate, $C(S)$, that he uses should be viewed as some polynomial-time evaluation or as a black-box predicate. He does not have to address this issue (his motivating example, SAT, satisfies the former but a parenthetical remark in his paper suggests the latter), as his results are valid either way and as he is improving the upper bound rather than establishing any lower bounds. In any case, note that in contrast to Simon's and the present paper's exponential superiority results, Grover's algorithm beats the obvious brute-force deterministic algorithm by a quadratic factor.

Note added in revision: Since this work was done, Chen and Diao [9] have claimed an exponential-type speedup for Grover-type searching, but with the iterations being dynamic and using auxiliary oracle functions. (Remark III.4 of their paper asserts without proof, with proof promised for the future, that gate growth rate will be such as to underpin a claim that the overall algorithmic complexity indeed is of the right growth rate.) We mention that their result is not an almost-everywhere hardness result. In fact, their paper does not discuss at all the lower bound, and to support an almost-everywhere hardness claim one would have to prove detailed results taking into account the fact that queries on one input might query oracle parts intended to apply to other inputs; this is exactly the type of argument that, in the different context of our problem, the present paper develops.

2.2. Limitations of Simon's Result

As described in Section 2.1, Simon–Brassard–Høyer show, for computing with respect to a black-box function, the inf nitely-often exponential superiority of exact quantum polynomial-time computing over classical deterministic computing (and even over classical bounded-error computing), on a particular problem. Since we will always speak of computing with respect to a black-box function that may have a promise, we will henceforth stop mentioning that and take it to be implicit from context as is standard in the literature.

Are there any worries or limitations to Simon's work? Simon (when one tightens his upper bound to QP via the work of Brassard–Høyer) gives an "inf nitely often" result: a problem that is in QP but such that each classical bounded-error machine solving the problem takes exponential time on inf nitely many inputs. However, "inf nitely many" says no more than it seems to. In fact, for Simon's problem, there are classical deterministic machines that solve the problem essentially instantly (i.e., in $n + 1$ steps on inputs of length n) on the vast majority of inputs—in fact, on all but one input of each length.

So, even though Simon proves inf nitely-often superiority, in fact for his problem the superiority occurs only on an exponentially thin portion of inputs. In contrast, the present paper achieves exponential superiority for exact quantum polynomial time on *all* suff ciently long inputs (so, for example, each classical machine for the problem will take subexponential time on at most a f nite set of inputs).

This is well motivated, as one issue of broad importance in the area of quantum computing is to gain an understanding of exactly what potential quantum computers hold, i.e., what superiority over classical computers they offer.

Thus, it is not surprising that the relation between classical and quantum computing is currently under intense scientif c scrutiny. We brief y mention some other works that have disclosed various facets of this relation and that exhibit, in different settings or different time classes, superiority in favor of quantum computing. As noted above, an early paper of Berthiaume and Brassard [6] raised the important issue of almost-everywhere hardness for quantum computing and showed that there are tasks that can be done in exact exponential time on quantum machines but on which each classical deterministic machine requires double exponential time almost everywhere. In contrast, our paper achieves almost-everywhere separation for exact quantum polynomial time and handles bounded-error as well as deterministic classical machines. Independent of the work of the present paper, which f rst appeared as [13], Ambainis and de Wolf studied average-case separations with respect to the uniform distribution [2, 3]. What is the relationship between their work and ours? Of course, almost-everywhere separation implies average-case separation in the standard sense, and thus our main result certainly implies average-case separation with respect to the uniform distribution. However, their paper is formally incomparable to ours as the models are exceedingly different (some ways in favor of the strength of their results, and some ways in favor of the strength of our results), for example (in their section related to this paper, their Section 4): (1) their fast quantum algorithms are Las Vegas-type algorithms (and thus some computation paths may take far longer than polynomial time) rather than exact quantum algorithms, (2) their input is exponentially long relative to their "n" and so they are actually distinguishing quantum logarithmic query complexity from classical polynomial query complexity, (3) we are computing a total (for the specif c oracle obtained in our proof) non-Boolean function and they are computing a total Boolean function (note that due to work of Beals et al. [4] it is known that in the query complexity model, which is the model of Beals et al. and of Ambainis and de Wolf but not of the present paper, superpolynomial query complexity gaps between quantum and classical computation cannot ever be obtained for total Boolean functions; but keep in mind that this does not speak directly to the issue of time complexity gaps in standard, non-(random access)-type-time-counting models), (4) their model of input and queries is different than ours as in some sense their input is their oracle (and so uniform distribution must be viewed in this context) and their notion (see also [4]) of query complexity essentially measures accessing the input itself, and (5) they study average-case complexity but we study almost-everywhere separations. Finally, we mention that in the important (but completely different) area of communication complexity, Raz [16] has shown that for promise problems there is an exponential gap between quantum communication complexity (which in particular is logarithmic on his problem) and classical probabilistic communication complexity (which he gives a lower-bound on as a root of the input size).

3. ALMOST-EVERYWHERE SUPERIORITY FOR QUANTUM POLYNOMIAL TIME

Let us start by explicitly stating where we will go. Recall that, as is common, we will throughout this paper denote quantum exact polynomial time (see [8]) by QP, though it sometimes in earlier papers is denoted EQP. Recall that what Simon's main theorem states (again, using here the Brassard–Høyer improvement of the upper bound to QP) is the following.

THEOREM 3.1 ([18, Theorem 3.4] augmented by [8]). *There is a constant $\epsilon > 0$ and a (function) oracle $\mathcal{O}$ relative to which there is a language B in exact quantum polynomial time such that each bounded-error classical Turing machine accepting B requires time more than $2^{\epsilon n}$ on infinitely many inputs.*

What we will prove is the following result, which extends the superiority from merely inf nitely often to instead almost everywhere.

THEOREM 3.2. *There is a constant $\epsilon > 0$ and a (function) oracle $\mathcal{O}$ relative to which there is a problem B computable in exact quantum polynomial time such that each bounded-error classical Turing machine computing B requires time more than $2^{\epsilon n}$ on all but a finite number of inputs.*

It follows immediately that this problem also demonstrates the almost-everywhere superiority of quantum computation over deterministic computation, when computing relative to a black-box function.

Some comments are in order regarding Theorem 3.2. First, we mention that the computational task on which we prove almost-everywhere exponential superiority for quantum computing is, in contrast with Simon's task, a function rather than a language. Second, note that bounded-error is a property of a machine in concert with its oracle, so in Theorem 3.2 and what follows "bounded-error machine" will always mean with respect to its oracle. Third, we should explicitly def ne what we mean by a probabilistic function.

DEFINITION 3.1. We say a function f is bounded-error Turing computable iff there is an $\epsilon > 0$ and a probabilistic Turing machine M such that, on each $x \in \Sigma^*$,

$$\text{Prob}(M(x) = f(x)) \geq 1/2 + \epsilon.$$

If M is a probabilistic Turing machine satisfying the above relation, we say that M is a bounded-error machine having error probability at most $1/2 - \epsilon$.

A given probabilistic machine is said to run in time t on input x if each of its computation paths completes in at most t steps.

Finally, we review a bit about Simon's result, as his result motivated our work, as we should credit him for the connections between his construction and ours, and as it is important to point out why the obvious transformation of his result does *not* give the result we seek.

The key construction used by Simon is described in the statement of the following result.

THEOREM 3.3 [18, THEOREM 3.3]. *Let $\mathcal{O}$ be a (function) oracle constructed as follows: for each n, a random n-bit string $s(n)$ and a random bit $b(n)$ are uniformly chosen from $\{0,1\}^n$ and $\{0,1\}$, respectively. If $b(n) = 0$, then the function $f_n : \{0,1\}^n \to \{0,1\}^n$ chosen for $\mathcal{O}$ to compute on n-bit queries is a random function uniformly distributed over permutations on $\{0,1\}^n$; otherwise it is a random function uniformly distributed over two-to-one functions such that $f_n(x) = f_n(x \oplus s(n))$ for all x, where $\oplus$ denotes bitwise exclusive-or. Then any* PTM *(probabilistic Turing machine) that queries $\mathcal{O}$ no more than $2^{n/4}$ times cannot correctly guess $b(n)$ with probability greater than $(1/2) + 2^{-n/2}$, over choices in the construction of $\mathcal{O}$.*[4]

Simon's "test language" that, based on this oracle, gives one the lower-bound of Theorem 3.1 is quite simply the issue of testing the bit described above, that is, the test language that is in QP but on which bounded-error $2^{\epsilon n}$-time classical Turing machines all err on inf nitely many inputs is $\{1^n \mid b(n) = 1\}$.

[4] The statement here is taken exactly from Simon. There are some informalities in Simon's statement—the fact that what independence is assumed is not explicitly stated and that the case "$b(n) = 1 \wedge s(n) = 0^n$" won't give an (exactly-2)-to-1 function.

It might be very tempting to exactly adopt the oracle $\mathcal{O}$ of Simon, but using instead of his test language the new test language: $\hat{L} = \{w \mid b(|w|) = 1\}$. This change attempts to "smear" the diff culty of 1^n onto all strings of length n, and even attempts to achieve the language analog of our desired result.

Unfortunately, this provably does not work. Why? A PTM can use the information in the input to (very rarely, but often enough) help it guess $s(n)$, in particular, certainly when it holds that both $b(n) = 1$ and the input happens to be $s(n)$.[5]

So, our construction takes a different tack. Intuitively speaking, the above problem should be removed if we increase the information content of the xor-bitmask well beyond that which input strings can give away. To achieve this, we double the information content of the xor-bitmask string and demand that our functions discover this string.

The rest of the paper concerns the proof of Theorem 3.2.

Proof of Theorem 3.2. We consider function oracles A of the following form: A is a collection of functions $(f_{n,A})_{n\in\mathbb{N}^+}$ with the following properties:

(i) $f_{n,A} : \{0, 1\}^n \to \{0, 1\}^{n-1}$,

(ii) $f_{n,A}$ is 2-to-1,

(iii) there is a string $s_{n,A}$ in $\{0, 1\}^n - \{0^n\}$ such that for all x of length n, $f_{n,A}(x \oplus s_{n,A}) = f_{n,A}(x)$.

Let $\mathcal{A}$ be the set of all such oracles. One can easily induce a probability measure on $\mathcal{A}$. Indeed, $\mathcal{A}$ is the product of the sets $(\mathcal{A}_i)_{i\in\mathbb{N}^+}$, where, for each $i \in \mathbb{N}^+$, $\mathcal{A}_i$ is the set of all functions f mapping $\{0, 1\}^i$ into $\{0, 1\}^{i-1}$ and having the properties (i), (ii), and (iii). On each set $\mathcal{A}_i$ we consider the probability measure given by the uniform distribution and then we consider the product measure on $\mathcal{A}$. This is identical to choosing, for each n independently, $f_{n,A}$ according to the uniform distribution over all functions with the properties (i), (ii), and (iii). All the probabilistic considerations that follow will be relative to this probability measure. It is important to observe that choosing $f_{n,A}$ uniformly at random amounts to the random selection of a string s from $\{0, 1\}^n$–$\{0^n\}$ and to the independent random selection of a permutation from $\{0, 1\}^{n-1}$ to $\{0, 1\}^{n-1}$ that dictates how the 2^{n-1} pairs $(u, u \oplus s)_{u\in\{0,1\}^n}$, ordered in some canonical way and identif ed with $\{0, 1\}^{n-1}$, are mapped into $\{0, 1\}^{n-1}$.

Let $A \in \mathcal{A}$. We def ne g_A, a function mapping, for all $n \geq 1$, strings of length n into strings of length $2n$, by

$$g_A(w) = s_{2|w|,A},$$

i.e., $g_A(w)$ is the unique string s with the property that for all x of length $2|w|$,

$$f_{2|w|,A}(x \oplus s) = f_{2|w|,A}(x).$$

It follows from the work of Brassard and Høyer [8] that there is a machine running in quantum polynomial time that computes g_A for all $A \in \mathcal{A}$.

Later in this proof, we will prove the following claim.

Claim 3.1. There is a set of oracles $\mathcal{B}_0$ having measure one in $\mathcal{A}$, such that for every $A \in \mathcal{B}_0$ and every deterministic oracle machine M the following holds: for almost every input w, M^A either runs for more than $2^{|w|/4} - 2$ steps or does not calculate $g_A(w)$.

[5] Just to be explicit here for absolute clarity, and assuming in light of the comments in Footnote 4 that we never allow the choice "$b(n) = 1 \wedge s(n) = 0^n$," consider the PTM that on each input w does:

{$n = |w|$; a = output of oracle $\mathcal{O}$ on input 0^n; b = output of oracle $\mathcal{O}$ on input $0^n \oplus w$; if $a = b$ and $w \notin 0^*$ then output "$b(|w|) = 1$ and $s(|w|) = w$" else output "$b(|w|) = 0$"}.

This machine will, on an inf nite number of inputs w (on each length n for which $b(n) = 1$, on the input that equals $s(n)$; and for each length n for which $b(n) = 0$, on all inputs), correctly determine $b(|w|)$ with probability one (relative to the choices of the PTM). Of course, this machine is not correctly accepting $\hat{L} = \{w \mid b(|w|) = 1\}$, but the machine is enough to show that keeping Simon's oracle $\mathcal{O}$ and just adopting the test set $\hat{L}$ does not establish Simon's Theorem 3.3 in the analogous case that applies here, i.e., where any length n string w may be the input. Note that the PTM given does substantially more than this; on each n with $b(n) = 1$, we have at least one input on which the PTM not only knows $b(n)$ but even discovers $s(n)$.

To move to bounded error probability machines, we invoke the techniques that Bennett and Gill [5] used to prove $P^A = BPP^A$ relative to a random oracle. An adaptation of their method shows the following.

Claim 3.2. There is a set of oracles $\mathcal{B}_1$ having measure one in $\mathcal{A}$, such that for any probabilistic oracle machine N and for any A in $\mathcal{B}_1$, there exists a deterministic oracle machine M with the following property: if N^A computes a function h with bounded error probability (in the sense of Def nition 3.1), then on all suff ciently long inputs w on which N^A runs in time $2^{|w|/10}$, $M^A(w) = h(w)$ and M^A runs in time $2^{|w|/4} - 2$

Claims 3.1 and 3.2 imply Theorem 3.2 (with $\mathcal{O} \in \mathcal{B}_0 \cap \mathcal{B}_1$, $\epsilon = 1/10$, and $B = g_{\mathcal{O}}$). For suppose for a contradiction that there exists a probabilistic oracle machine N and $\mathcal{O} \in \mathcal{B}_0 \cap \mathcal{B}_1$, such that $N^{\mathcal{O}}$ bounded-error computes $g_{\mathcal{O}}$ in the sense of Def nition 3.1 and such that $N^{\mathcal{O}}$ runs in time $2^{|w|/10}$ for inf nitely many inputs w. Then, by Claim 3.2, there exists a deterministic oracle machine M that, for inf nitely many inputs w, calculates $g_{\mathcal{O}}(w)$ and runs in time $2^{|w|/4} - 2$. But that contradicts Claim 3.1.

For completeness and since there are some differences between our context and the one in the paper of Bennett and Gill [5], we will prove Claim 3.2 in detail. In the proof, we will assume that all the oracles A are in $\mathcal{A}$. If N is a probabilistic oracle machine, $A \in \mathcal{A}$ an oracle, and N^A computes a function h with bounded error probability, we will write $N^A(w)$ to denote $h(w)$.

Let N be a probabilistic oracle machine, let A be an oracle (in $\mathcal{A}$), and let r be a rational number such that $0 \le r < 1/2$ and N^A computes a function with error probability at most r. Let us f x, as a parameter, a positive integer k.

If we iterate N on input w a polynomial number of times (the polynomial depends on k and r), and, on each computation path, output the majority output among the polynomially many computations of N if a majority output exists (if not, we (arbitrarily) output 0), we get a new machine $N'^A_{k,r}$ that, on all oracles A on which N has error probability at most r, for every input w for which every computation path of $N^A(w)$ terminates, computes the same function as N^A but with error probability at most $(1/k)2^{-(2|w|+1)}$.

For all oracles A, $N'^A_{k,r}$ runs in time $2^{|w|/9}$ on all suff ciently long inputs w on which N^A is running in time $2^{|w|/10}$. Also note that $N'^A_{k,r}$ queries strings of length at most $2^{|w|/10}$ on all the inputs w on which N^A runs in time $2^{|w|/10}$. From $N'_{k,r}$, we build a deterministic machine $M_{N,k,r}$ as follows. Machine $M_{N,k,r}$ on input w simulates $N'_{k,r}$ on input w and each time N' requires a random bit for doing a probabilistic step, $M_{N,k,r}$ takes this bit to be the f rst bit of $f_{t,A}(0^t)$, where t is the smallest integer $> 2^{|w|/10}$ such that 0^t has not been queried before during the simulation on input w. It is easy to check that for all strings w that are long enough, if $N'^A_{k,r}$ on w runs in time $2^{|w|/9}$, then $M^A_{N,k,r}$ on w runs in time $2^{|w|/4} - 2$. For each w, and each rational r with $0 \le r < 1/2$, let $E_{N,k,r,w}$ be the class of oracles A on which N^A on input w runs in $2^{|w|/10}$ steps and has error probability at most r, and on which $M^A_{N,k,r}(w) \ne N'^A_{k,r}(w)$. Let $U_1, \ldots, U_s$ be all the partial functions def ned on the strings of length at most $2^{|w|/10}$ such that for all $i \in \{1, \ldots, s\}$, N^{U_i} on input w runs in $2^{|w|/10}$ steps with error probability at most r. For an oracle A, let A_{low} denote its restriction to the strings of length at most $2^{|w|/10}$. Then

$$\mathrm{Prob}_A(A \in E_{N,k,r,w}) = \sum_{i=1}^{s} \mathrm{Prob}_A\big(M^A_{N,k,r}(w) \ne N'^A_{k,r}(w) \,\big|\, A_{low} = U_i\big) \cdot \mathrm{Prob}_A(A_{low} = U_i).$$

Now, $\mathrm{Prob}_A(M^A_{N,k,r}(w) \ne N'^A_{k,r}(w) \mid A_{low} = U_i)$ is the probability that $M^A_{N,k,r}(w) \ne N'^A_{k,r}(w)$ given that the regular queries of both machines are answered according to U_i. Since the only queries besides those stipulated by U_i that are involved in the conditioned event "$M^A_{N,k,r}(w) \ne N'^A_{k,r}(w)$" are those used by $M_{N,k,r}$ to simulate the random bits used by $N'^{U_i}_{k,r}$ on w, it follows that the above conditioned probability is the error probability of $N'^{U_i}_{k,r}(w)$ which is at most $(1/k)2^{-(2|w|+1)}$. It follows that $\mathrm{Prob}_A(A \in E_{N,k,r,w}) \le (1/k)2^{-(2|w|+1)} \cdot \sum_{i=1}^{s} \mathrm{Prob}_A(A_{low} = U_i) \le (1/k)2^{-(2|w|+1)}$.

Let $E_{N,k,r}$ denote the set $\cup_w E_{N,k,r,w}$, where the union is taken over all strings w. Note f rst that if $A \notin E_{N,k,r}$ and if N^A has error probability at most r, then $M^A_{N,k,r}(w) = N^A(w)$ on all inputs w on which N^A runs in $2^{|w|/10}$ steps. We have that $\mathrm{Prob}_A(A \in E_{N,k,r}) \le \sum_w \mathrm{Prob}_A(A \in E_{N,k,r,w}) \le (1/k)\sum_w 2^{-(2|w|+1)} = 1/k$. Therefore the measure of $\cap_{k\ge1} E_{N,k,r}$ is zero and thus the measure of $\mathcal{A} - \cap_{k\ge1} E_{N,k,r}$ is one. We take $\mathcal{B}_1 = \cap_{N,r}(\mathcal{A} - \cap_{k\ge1} E_{N,k,r})$, where the f rst intersection is taken over

all probabilistic Turing machines N and rationals r such that $0 \leq r < 1/2$. The set $\mathcal{B}_1$ has measure one because it is a countable intersection of sets of measure one.

Let N be a probabilistic Turing machine and A be an oracle in $\mathcal{B}_1$ such that N^A has bounded error probability at most r for rational r with $0 \leq r < 1/2$. It follows that $A \in \mathcal{A} - E_{N,k,r}$ for some k. On all suff ciently long inputs w on which N^A runs in $2^{|w|/10}$ steps, $M_{N,k,r}$ runs in time $2^{|w|/4} - 2$ and $M^A_{N,k,r}(w) = N^A(w)$. This completes the proof of Claim 3.2.

We now prove Claim 3.1, that is, we have to show that there is a set of oracles $\mathcal{B}_0$ having measure one in $\mathcal{A}$, such that for every $A \in \mathcal{B}_0$ and every deterministic oracle machine M the following holds: for almost every input w, M^A either runs for more than $2^{|w|/4} - 2$ steps or does not calculate $g_A(w)$.

Thus, let M be a deterministic oracle machine that attempts to calculate g_A. We modify M so that at the end of its computation, having a string s on its output tape, it asks the oracle A for the values of $f_{|s|,A}(0^{|s|})$ and $f_{|s|,A}(s)$. Let M' be the modif ed machine. The reason for this modif cation is so we are sure there is a "collision" if M has the correct string s, as we will now make formal and clear. We say that for an oracle A, two distinct strings x and y collide if $f_{|x|,A}(x) = f_{|y|,A}(y)$. Let us f x an input w and let $n = |w|$. Observe that

$$\begin{aligned}&\mathrm{Prob}_A\big(M^A \text{ runs at most } 2^{n/4} - 2 \text{ steps and calculates } g_A(w)\big)\\ &\quad\leq \mathrm{Prob}_A\big(M'^A \text{ queries at most } 2^{n/4} \text{ strings and two queried strings}\\ &\qquad\text{of length } 2n \text{ collide with respect to } A\big),\end{aligned} \tag{1}$$

because if M^A is correct on w, then M'^A at the end of its computation will ask 0^{2n} and $s_{2n,A}$, and these two strings will collide.

We assume without loss of generality that for each z and for each oracle A it holds that $M'^A(z)$ does not query the same string twice during its run. Let $x_1, x_2, \ldots, x_k$ be, in the order in which they are queried, the strings that M' queries on input w. Of course, k and the set of strings are random variables (in other words they depend on the oracle A). We will show the following fact.

Fact 3.1. $p_w =_{def} \mathrm{Prob}_A(k \leq 2^{|w|/4}$ and there is a collision for a pair of strings of length $2|w|$ in $\{x_1, \ldots, x_k\}) \leq 2^{-1.4|w|}$.

Assuming that the fact holds, we have

$$\sum_{w\in\{0,1\}^*} p_w = \sum_{\ell=0}^{\infty} \sum_{w\in\{0,1\}^\ell} p_w = \sum_{\ell=0}^{\infty} 2^\ell \cdot 2^{-1.4\ell} = \sum_{\ell=0}^{\infty} 2^{-0.4\ell} < \infty. \tag{2}$$

By the Borel-Cantelli Lemma and taking into account (1) it follows that

$$\mathrm{Prob}_A\big(\text{for inf nitely many inputs } w,\ M^A(w) \text{ makes at most } 2^{|w|/4} - 2 \text{ steps and computes } g_A(w)\big) = 0.$$

Since there are a countable number of deterministic oracle machines M, we obtain that

$$\begin{aligned}&\mathrm{Prob}_A\big(\text{there exists } M \text{ that, on inf nitely many inputs } w, \text{ runs at most } 2^{|w|/4}\\ &\quad - 2 \text{ steps and that computes } g_A(w)\big) = 0.\end{aligned}$$

Consequently,

$$\begin{aligned}&\mathrm{Prob}_A\big(\text{for all } M,\ M^A, \text{ on almost every input } w, \text{ either runs more than } 2^{|w|/4}\\ &\quad - 2 \text{ steps or does not compute } g_A(w)\big) = 1,\end{aligned} \tag{3}$$

which is the desired assertion.

We still must prove Fact 3.1. In this proof, for brevity, collisions will always refer to strings of length $2n$ and will always be with respect to the oracle A. We will drop a subscript from the functions f, with the understanding that the missing subscript is equal to the length of the argument. We will also write Prob(...) for $\mathrm{Prob}_A(\ldots)$ when this is clear from the context.

Decomposing the event "$k \leq 2^{n/4}$ and collision in $\{x_1, \ldots, x_k\}$" into mutually disjoint events, we have

$$\begin{aligned}
&\mathrm{Prob}\big(k \leq 2^{n/4} \text{ and collision in } \{x_1, \ldots, x_k\}\big) \\
&\quad = \mathrm{Prob}\big(k \leq 2^{n/4} \text{ and collision in } \{x_1, x_2\}\big) \\
&\qquad + \mathrm{Prob}\big(k \leq 2^{n/4} \text{ and } x_3 \text{ collides with } x_1 \text{ or } x_2 \text{ and no collision in } \{x_1, x_2\}\big) \\
&\qquad + \cdots + \mathrm{Prob}\big(k \leq 2^{n/4} \text{ and } x_k \text{ collides with } x_1 \text{ or } x_2 \text{ or } \ldots \text{ or } x_{k-1} \\
&\qquad\quad \text{and no collision in } \{x_1, \ldots, x_{k-1}\}\big) \\
&\quad \leq \sum_{j=2}^{2^{n/4}} \mathrm{Prob}(x_j \text{ collides with } x_1 \text{ or } x_2 \text{ or } \ldots \text{ or } x_{j-1} \text{ and no collision in } \{x_1, \ldots, x_{j-1}\}), \qquad (4)
\end{aligned}$$

with the convention that events involving some x_j with $j > k$ are empty (and thus have probability zero). We look at the general term in the above sum.

$$\begin{aligned}
&\mathrm{Prob}(x_j \text{ collides with } x_1 \text{ or } x_2 \text{ or } \ldots \text{ or } x_{j-1} \text{ and no collision in } \{x_1, \ldots, x_{j-1}\}) \\
&\quad = \sum \mathrm{Prob}(x_j \text{ collides with } x_1 \text{ or } \ldots \text{ or } x_{j-1} \text{ and no collision in } \{x_1, \ldots, x_{j-1}\} \mid \\
&\qquad (\forall i \in \{1, \ldots, j\})[x_i = u_i] \text{ and } (\forall i \in \{1, \ldots, j-1\})[f_A(u_i) = a_i]) \\
&\qquad \times \mathrm{Prob}((\forall i \in \{1, \ldots, j\})[x_i = u_i] \text{ and } (\forall i \in \{1, \ldots, j-1\})[f_A(u_i) = a_i]), \qquad (5)
\end{aligned}$$

where the sum is taken over all j-tuples $(u_1, \ldots, u_j)$ of distinct strings in $\{0, 1\}^*$ (that we consider as potential queries of M' on w) and over all possible answers $(a_1, \ldots, a_{j-1})$ to the queries $u_1, \ldots, u_{j-1}$ such that the possible answers of length $2n - 1$ are distinct (these are answers to queries of length $2n$ and they are distinct because there is no collision in $\{u_1, \ldots, u_{j-1}\}$). Let us fix a tuple $(u_1, \ldots, u_j)$ of possible distinct queries and a tuple $(a_1, \ldots, a_{j-1})$ of possible answers as above and let us consider the probability

$$\begin{aligned}
&\mathrm{Prob}(x_j \text{ collides with } x_1 \text{ or } \ldots \text{ or } x_{j-1} \text{ and no collision in } \{x_1, \ldots, x_{j-1}\} \mid \\
&\qquad (\forall i \in \{1, \ldots, j\})[x_i = u_i] \text{ and } (\forall i \in \{1, \ldots, j-1\})[f_A(u_i) = a_i]),
\end{aligned}$$

which is of course equal to

$$\begin{aligned}
&\mathrm{Prob}(u_j \text{ collides with } u_1 \text{ or } \ldots \text{ or } u_{j-1} \text{ and no collision in } \{u_1, \ldots, u_{j-1}\} \mid (\forall i \in \{1, \ldots, j\}) \\
&\quad [x_i = u_i] \text{ and } (\forall i \in \{1, \ldots, j-1\})[f_A(u_i) = a_i]). \qquad (6)
\end{aligned}$$

Note that the condition "no collision in $\{u_1, \ldots, u_{j-1}\}$" is subsumed by the condition "$(\forall i \in \{1, \ldots, j-1\})[f_A(u_i) = a_i]$" because the answers a_i, for $i = 1, \ldots, j-1$, are distinct with respect to those of them of length $2n - 1$. The conditions $f_A(u_i) = a_i$, for $i = 1, \ldots, j-1$, completely determine whether it is the case that for all $i \in \{1, \ldots, j\}$, the ith query is u_i, i.e., whether for all $i \in \{1, \ldots, j\}$, $x_i = u_i$. Thus the event {no collision in $\{u_1, \ldots, u_{j-1}\}$ and $(\forall i \in \{1, \ldots, j\})[x_i = u_i]$ and $(\forall i \in \{1, \ldots, j-1\})[f_A(u_i) = a_i]$} is either empty or is equal to the event {$(\forall i \in \{1, \ldots, j-1\})$ $[f_A(u_i) = a_i]$}. If it is empty, the probability in Eq. (6) is zero (by the standard convention regarding conditional probabilities). In the other case, the probability in Eq. (6) is equal to

$$\begin{aligned}
&\mathrm{Prob}(u_j \text{ collides with } \{u_1, \ldots, u_{j-1}\} \mid (\forall i \in \{1, \ldots, j-1\})[f_A(u_i) = a_i]) \\
&\quad = \frac{\mathrm{Prob}(u_j \text{ collides with } \{u_1, \ldots, u_{j-1}\} \text{ and } (\forall i \in \{1, \ldots, j-1\})[f_A(u_i) = a_i])}{\mathrm{Prob}((\forall i \in \{1, \ldots, j-1\})[f_A(u_i) = a_i])}.
\end{aligned}$$

If $|u_j| \neq 2n$ the above conditional probability is zero. So, we will consider that $|u_j| = 2n$. Let $U = \{u_i \mid i \in \{1, \ldots, j-1\} \text{ and } |u_i| = |u_j| = 2n\}$ and let $W = \{u_1, \ldots, u_{j-1}\} \setminus U$. Note that $\|U\|$, the

cardinality of U, is at most $j-1$. Observe also that u_j cannot collide with elements from W and that the events "u_j collides with some element in U and $f_A(u_i) = a_i$, for all u_i in U" and "$f_A(u_i) = a_i$, for all u_i in W" are independent. The events "$f_A(u_i) = a_i$, for all u_i in U" and "$f_A(u_i) = a_i$, for all u_i in W" are also independent (the choices made in the construction of the oracle at different lengths are independent). Therefore the probabilities involving strings $u \in W$ cancel and it remains to evaluate

$$\frac{\text{Prob}(u_j \text{ collides with } \{u_i \mid u_i \in U\} \text{ and } (\forall u_i \in U)[f_A(u_i) = a_i])}{\text{Prob}((\forall u_i \in U)[f_A(u_i) = a_i])}. \tag{7}$$

The events in the above equation depend on the choices of the string s and of the permutation that determines $f_{2n,A}$, and these two choices are independent, as we have observed when we built the probability measure. Let us focus on the event appearing in the numerator. For this event to hold, the string s, which is responsible for the collisions, must be chosen so as to make u_j collide with one of $\{u_i \mid u_i \in U\}$, and so as to prevent any collision in U (because the "answers" a_i to the "queries" u_i in U are distinct). If we f x one such string s, the 2^{2n-1} pairs $(u, u \oplus s)_{u \in \{0,1\}^{2n}}$ are determined, and the permutation def ning A at length $2n$ must be chosen so as to map u_i to a_i for all $u_i \in U$. The number of such permutations does not depend on the f xed string s. Thus, the numerator is equal to the probability over A that $s_{2n,A}$ is in the set $\{u_j \oplus u_i \mid u_i \in U\} \setminus \{u \oplus v \mid u, v \in U \text{ and } u \neq v\}$ times the probability that (for f xed s) the permutation def ning A at length $2n$ is compatible with $f_A(u_i) = a_i$, $u_i \in U$ (a probability that as noted above is the same for each s that is not of the form $u \oplus v$, $u, v \in U$, $u \neq v$). The f rst factor is at most

$$\frac{\|U\|}{2^{2n}-1} \leq \frac{j-1}{2^{2n}-1}.$$

Similarly, the denominator in Eq. (7) is equal to the probability that s is a string of length $2n$ different from 0^{2n} and not in the set $\{u \oplus v \mid u, v \in U \text{ and } u \neq v\}$ times the probability that (for f xed s) the permutation def ning A at length $2n$ is compatible with $f_A(u_i) = a_i, u_i \in U$ (and thus the second factor of the denominator is the same as the second factor of the numerator). The f rst factor of the denominator is at least

$$\frac{2^{2n}-1-\|U\|(\|U\|-1)/2}{2^{2n}-1} \geq \frac{2^{2n}-1-(j-1)(j-2)/2}{2^{2n}-1}.$$

Consequently, the fraction in Eq. (7) is bounded from above by

$$\frac{j-1}{2^{2n}-1-\frac{(j-1)(j-2)}{2}}.$$

Substituting in Eq. (5), we have that:

$$\begin{aligned}
&\text{Prob}(x_j \text{ collides with } x_1 \text{ or } x_2 \text{ or } \ldots \text{ or } x_{j-1} \text{ and no collision in } \{x_1, \ldots, x_{j-1}\}) \\
&\leq \tfrac{j-1}{2^{2n}-1-\frac{(j-1)(j-2)}{2}} \textstyle\sum \text{Prob}((\forall i \in \{1, \ldots, j\})[x_i = u_i] \text{ and } \forall i \in \{1, \ldots, j-1\})[f_A(u_i) = a_i]) \\
&\leq \tfrac{j-1}{2^{2n}-1-\frac{(j-1)(j-2)}{2}}.
\end{aligned}$$

Thus, returning to Eq. (4), we obtain that, for $n \geq 4$,

$$\begin{aligned}
&\text{Prob}\big(k \leq 2^{n/4} \text{ and collision in } \{x_1, \ldots, x_k\}\big) \\
&\leq \sum_{j=2}^{2^{n/4}} \frac{j-1}{2^{2n}-1-\frac{(j-1)(j-2)}{2}} \leq \sum_{j=2}^{2^{n/4}} \frac{2^{n/4}}{2^{2n}-1-(2^{n/2}-1)} \\
&\leq \frac{2^{n/2}}{2^{2n}-2^{n/2}} = \frac{1}{2^{3n/2}-1} \leq \frac{1}{2^{1.4n}},
\end{aligned}$$

For $n < 4$, the above probability is 0, and so certainly is less than $\frac{1}{2^{1.4n}}$. This ends the proof of Fact 3.1. ■

We mention that though it sometimes happens in complexity theory that function results immediately yield corresponding language results, it is not the case that our main result implies, at least in any obvious way, the corresponding language result. Let us be more explicit on this point. One might well wonder:

> It seems that your function result will easily give the analogous language result. Why? Basically, by using the standard way we coerce function complexity into language complexity, i.e., via making a language that slices out bits or that pref x searches. For example, using the f rst of these approaches, take your hard function, call it g. Now consider the function h def ned as $h(\langle y, i\rangle) =$ the ith bit of $g(y)$. Since g truth-table reduces to h, it follows that if h has fast algorithms then g has fast algorithms (the relation depending on the length of the query strings and the number of queries, but in fact in our case these are such that one could make a good claim). But you prove/claim that g does not have fast classical bounded-error algorithms, so neither can h. And certainly (this actually is the case) Brassard–Høyer easily still gives us that h is quantum-easy to compute.

However, this reasoning is not valid. The above argument would be f ne if we were dealing with inf nitely-often hardness. However, we are seeking to prove almost-everywhere hardness, and in fact the bit-slices of an a.e.-hard function are not necessarily a.e.-hard. To see this, consider any a.e.-hard function and pref x a 1 to all its outputs. This is still a.e.-hard but its bit-slices are inf nitely often trivial, namely, the f rst bit of each output is 1. Of course, our hard function does not seem to have any such "obvious" or easy bits, but this is just an informal, tempting hope rather than a valid proof.[6]

Another observation is that the proof of Theorem 3.2 actually shows the following stronger result.

THEOREM 3.4. *There is a constant $\epsilon > 0$ and a function oracle $\mathcal{O}$ relative to which there is a problem B computable in exact quantum polynomial time such that if M is any bounded-error classical Turing machine, then on all but a finite number of inputs w on which the machine correctly solves B, M requires more than $2^{\epsilon|w|}$ steps.*

In other words, even classical machines that are allowed to err inf nitely many times in their computation of the problem B still need more than $2^{\epsilon n}$ time on almost every input on which they are correct. The result follows immediately from Eq. (3) and from the simulation of bounded-error machines by deterministic machines, both relativized with a random oracle, via the Bennett–Gill technique.

ACKNOWLEDGMENT

We thank Andris Ambainis, Gilles Brassard, Peter Høyer, Ronald de Wolf, and the anonymous referees for valuable literature pointers, comments, and advice.

REFERENCES

1. Allender, E., Beigel, R., Hertrampf, U., and Homer, S. (1993), Almost-everywhere complexity hierarchies for nondeterministic time, *Theoret. Comput. Sci.* **115**, 225–241.
2. Ambainis, A., and de Wolf, R. (1999), "Average-Case Quantum Query Complexity," Technical Report quant-ph/9904079v2, Los Alamos e-Print Quantum Physics Technical Report Archive, November 11, 1999, *available at* http://xxx.lanl.gov/.
3. Ambainis, A., and de Wolf, R. (2000), Average-case quantum query complexity, *in* "Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science," Lecture Notes in Computer Science, Vol. 1770, pp. 133–144, Springer-Verlag, Berlin.
4. Beals, R., Buhrman, H., Cleve, R., Mosca, M., and de Wolf, R. (1998), Quantum lower bounds by polynomials, *in* "Proceedings of the 39th IEEE Symposium on Foundations of Computer Science," pp. 352–361, IEEE Comput. Soc., Los Alamitos, CA.
5. Bennett, C., and Gill, J. (1981), Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \mathrm{coNP}^A$ with probability 1, *SIAM J. Comput.* **10**, 96–113.
6. Berthiaume, A., and Brassard, G. (1994), Quantum oracle computing, *J. Modern Opt.* **41**, 2521–2535.
7. Boyer, M., Brassard, G., Høyer, P., and Tapp, A. (1998), Tight bounds on quantum searching, *Fortschritte Der Physik* **46**, 493–506.
8. Brassard, G., and Høyer, P. (1997), An exact quantum polynomial-time algorithm for Simon's problem, *in* "Proceedings of the 5th Israeli Symposium on Theory of Computing and Systems," pp. 12–23, IEEE Comput. Soc., Los Alamitos, CA.

[6] Note added in proof: New work of Hemaspaandra, Hemaspaandra, and Zimand ("Almost-everywhere superiority for quantum polynomial-time languages," University of Rochester Computer Science technical report TR-754, 2001) studies the language case.

9. Chen, G., and Diao, Z. (2000), "An Exponentially Fast Quantum Search Algorithm," Technical Report quant-ph/0011109, Los Alamos e-Print Quantum Physics Technical Report Archive, *available at* http://xxx.lanl.gov/.
10. Geske, J., Huynh, D., and Seiferas, J. (1991), A note on almost-everywhere-complex sets and separating deterministic-time-complexity classes, *Inform. and Comput.* **92**, 97–104.
11. Geske, J., Huynh, D., and Selman, A. (1987), A hierarchy theorem for almost everywhere complex sets with application to polynomial complexity degrees, *in* "Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science," Lecture Notes in Computer Science, Vol. 247, pp. 125–135, Springer-Verlag, Berlin.
12. Grover, L. (1996), A fast quantum mechanical algorithm for database search, *in* "Proceedings of the 28th ACM Symposium on Theory of Computing," pp. 212–219.
13. Hemaspaandra, E., Hemaspaandra, L., and Zimand, M. (1999), "Almost-Everywhere Superiority for Quantum Computing," Technical Report quant-ph/9910033, October 8, 1999, revised October 20, 1999, Los Alamos e-Print Quantum Physics Technical Report Archive, *available at* http://xxx.lanl.gov/.
14. Høyer, P. (1999), Personal communication.
15. Mihara, T., and Sung, S. (1998), A quantum polynomial algorithm in the worst case for Simon's problem, *in* "Proceedings of the 9th International Symposium on Algorithms and Computation," Lecture Notes in Computer Science, Vol. 1533, pp. 229–236. Springer-Verlag, Berlin.
16. Raz, R. (1999), Exponential separation of quantum and classical communication complexity, *in* "Proceedings of the 31st ACM Symposium on Theory of Computing," pp. 358–367, Assoc. Comput. Mach., New York.
17. Shor, P. (1997), Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comput.* **26**(5), 1484–1509.
18. Simon, D. (1997), On the power of quantum computation, *SIAM J. Comput.* **26**(5), 1474–1483.